

# ΣΧΕΔΙΑΣΜΌΣ ΣΥΣΤΗΜΆΤΩΝ VLSI

Δεύτερη Εργαστηριακή Άσκηση

Γώργος Ντάκος 1059569

26 Μαρτίου 2021

### Περίληψη

Στο παρακάτω κείμενο παραδίδεται η αναφορά της 2<sup>ης</sup> εργαστηριακής άσκησης του μαθήματος Σχεδιασμός Συστημάτων VLSI. Όλοι οι κώδικες έχουν συγγραφεί σε γλώσσα **VHDL** μέσω του **Notepad++** και έχουν μεταγλωττιστεί και επιβεβαιωθεί οι λειτουργίες τους μέσω του εργαλείου **ModelSim**.

## Περιεχόμενα

Ενότητα Β	4
Ενότητα Γ	7
Κώδικες σε VHDL	10

## Κατάλογος Σχημάτων

1	Πολυπλέκτης 2 σε 1 . . . . .	4
2	Simulation του πρώτου κώδικα του πολυπλέκτη . . . . .	5
3	Simulation του δεύτερου κώδικα του πολυπλέκτη . . . . .	6
4	Σχηματικό της λογικής συνάρτησης 1 . . . . .	7
5	Simulation του πρώτου κώδικα της λογικής συνάρτησης 1 . . . . .	8
6	Simulation του δεύτερου κώδικα της λογικής συνάρτησης 1 . . . . .	9

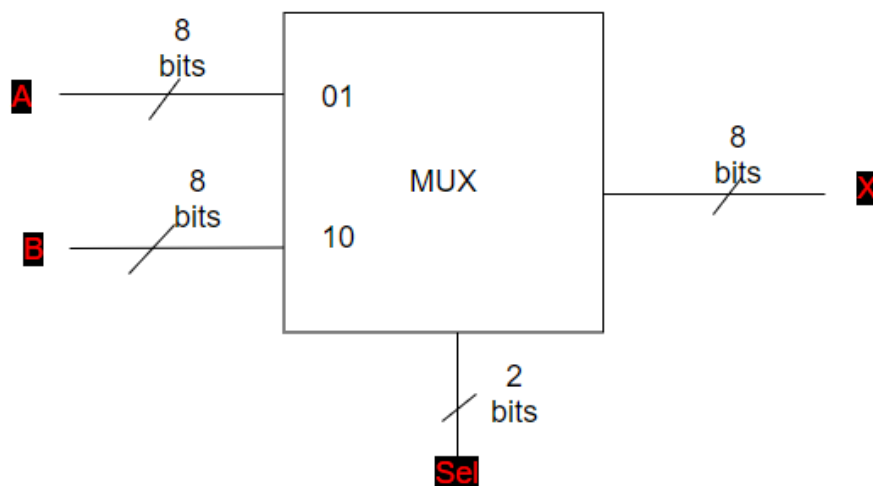
## Κατάλογος Πινάκων

1	Πίνακας αληθείας του πολυπλέκτη . . . . .	4
2	Πίνακας αληθείας της λογικής συνάρτησης 1 . . . . .	7

## Ενότητα B

### B.1,B.2

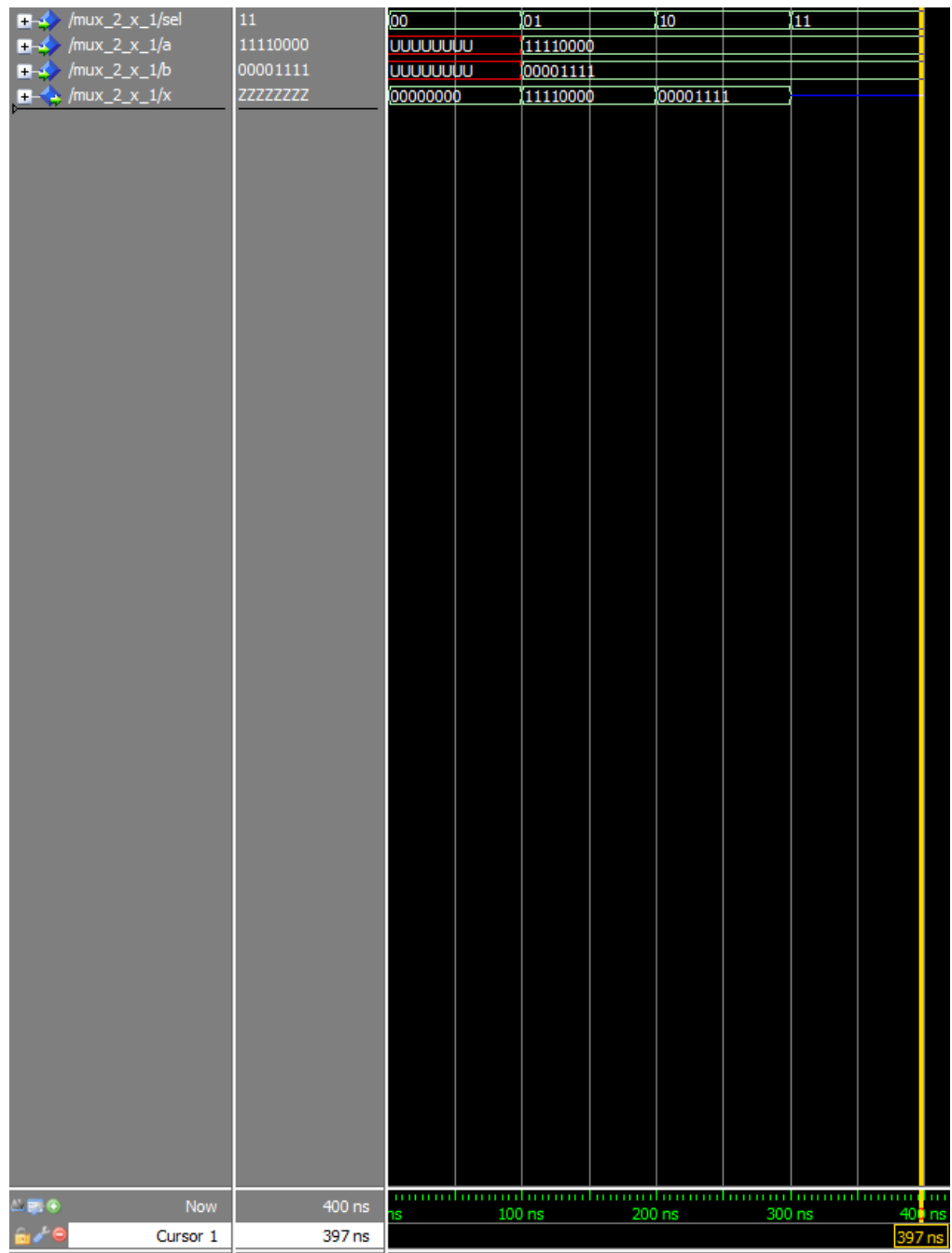
Στην εικόνα 1 παρατηρούμε ότι θέλουμε να φτιάξουμε τους κώδικες που να υλοποιούν 1 πολυπλέκτη 2 σε 1 των 8 bits η κάθε είσοδος και έξοδος με το σήμα sel να είναι των 2 bits . Η λειτουργία του κυκλώματος αυτού φαίνεται στον πίνακα αληθείας 1. Αφού κάνουμε compile και τους 2 κώδικες που μας ζητάει η άσκηση(διαφέρουν σε απλές βασικές εντολές) τους εξομοιώνουμε και παίρνουμε τα αποτελέσματα των εικόνων 2,3 όπου και διαπιστώνουν την ορθή λειτουργία τους.



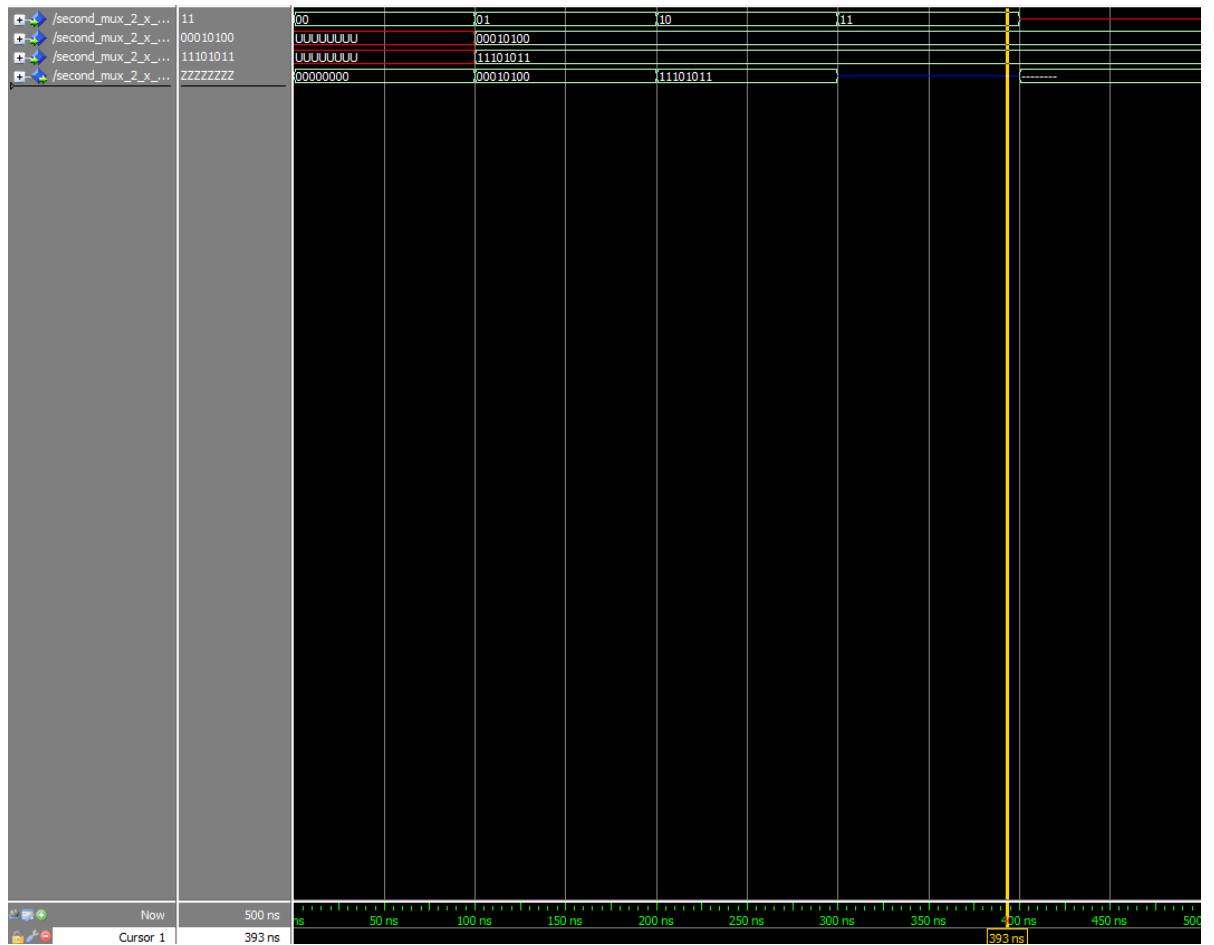
Σχήμα 1: Πολυπλέκτης 2 σε 1

sel	x
"00"	"00000000"
"01"	a
"10"	b
"11"	"ZZZZZZZZ"

Πίνακας 1: Πίνακας αληθείας του πολυπλέκτη



Σχήμα 2: Simulation του πρώτου κώδικα του πολυπλέκτη



Σχήμα 3: Simulation του δεύτερου κώδικα του πολυπλέκτη

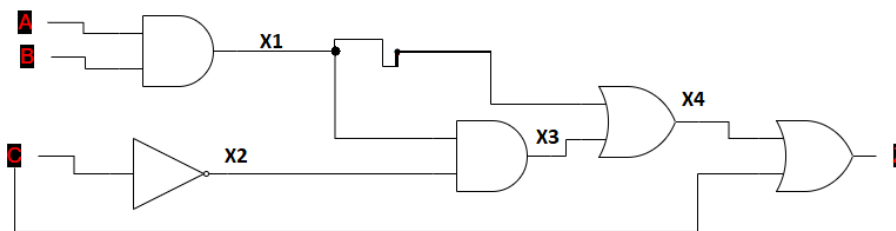
## Ενότητα Γ

### Γ.1,Γ.2

Η συγκεκριμένη άσκηση μας ζητάει να δημιουργήσουμε 2 κώδικες, ο πρώτος με προσέγγιση με βάση διεργασία ενώ ο δεύτερος με προσέγγιση structural δομής, για την παρακάτω λογική συνάρτηση:

$$Z = (A \wedge B \wedge \neg C) \vee (A \wedge B) \vee C \quad (1)$$

Η λογική συνάρτηση 1 σχηματικά φαίνεται στην εικόνα 4 και η λειτουργία της φαίνεται στον πίνακα αληθείας 2.



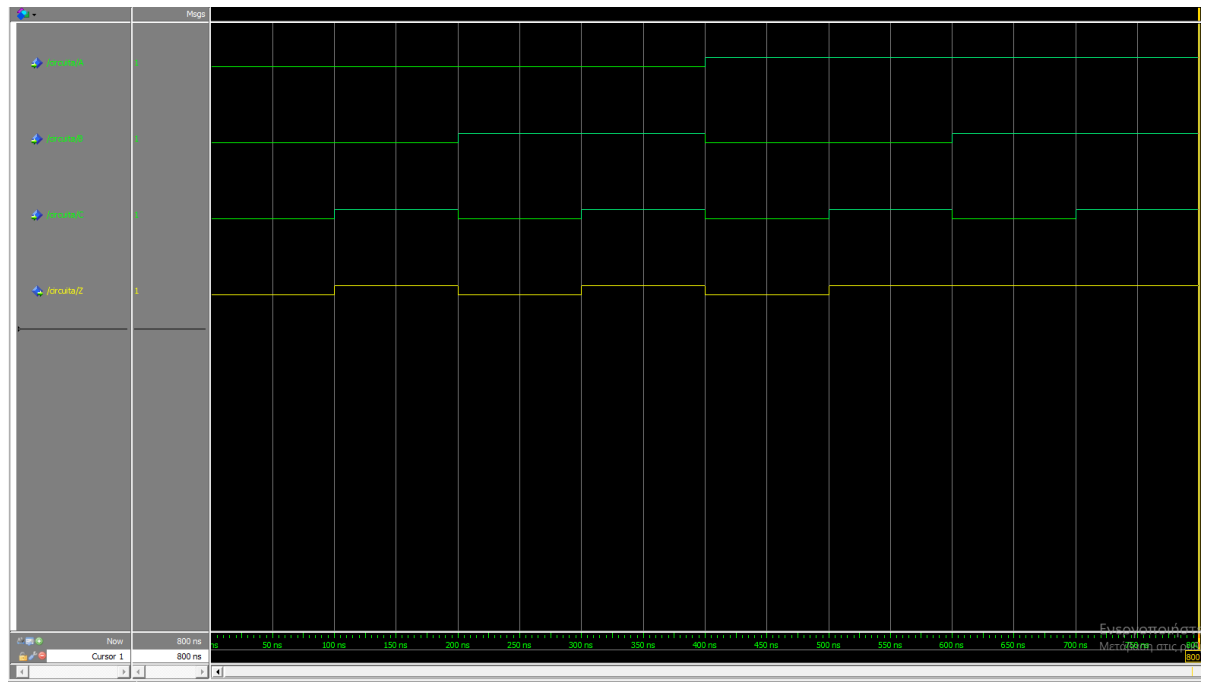
Σχήμα 4: Σχηματικό της λογικής συνάρτησης 1

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Πίνακας 2: Πίνακας αληθείας της λογικής συνάρτησης 1

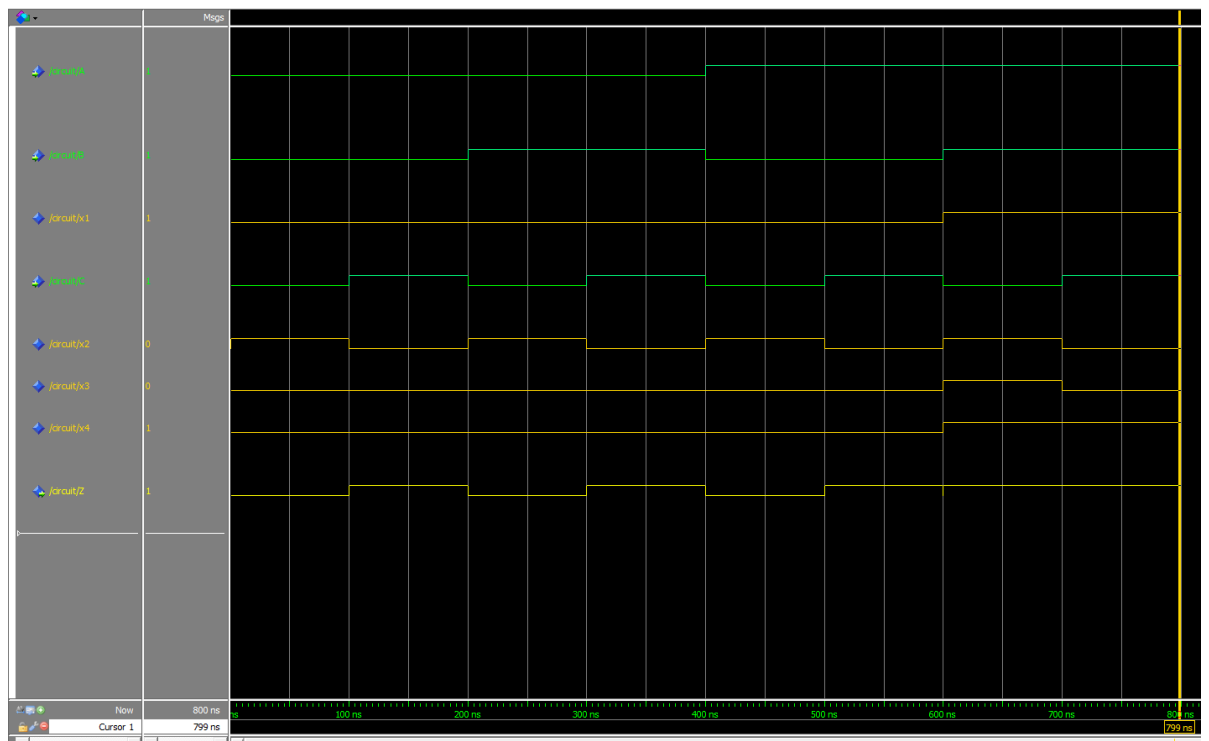
### Γ.3

Αφού κάνουμε compile και τους 2 κώδικες που μας ζητάει η άσκηση τους εξομοιώνουμε και παίρνουμε τα αποτελέσματα των εικόνων 5,6 όπου και διαπιστώνουν την ορθή λειτουργία τους.



Σχήμα 5: Simulation του πρώτου κώδικα της λογικής συνάρτησης 1





Σχήμα 6: Simulation του δεύτερου κώδικα της λογικής συνάρτησης 1

## Γ.4

Σύμφωνα από τους δύο κώδικες προφανώς και θα υπάρξουν διαφορές. Πρώτα πόλλα στον πρώτο κώδικα της λογικής συνάρτησης **1** το μπλοκ κώδικα μέσα στην εντολή **PROCESS** εκτελείται με ακολουθιακό τρόπο δηλαδή σειριακά σε αντίθεση με το δεύτερο κώδικα όπου και εκτελείται όλος παράλληλα. Τώρα στο **Simulation** δεν υπάρχει κάποια διαφορά πέρα από τα ενδιάμεσα σήματα, στο **Simulation** του δεύτερου κώδικα, τα οποία θα μπορούσαν και να μην υπάρχουν στο παράθυρο των κυματομορφών. Από την άλλη βέβαια ο **Synthesizer** στην πρώτη περίπτωση θα συνθέσει ένα οποιοδήποτε κύκλωμα το οποίο απλά θα ικανοποιεί τις συνθήκες μέσα στην εντολή **PROCESS** και όχι απαραίτητα αυτό της εικόνας **4**, ενώ στο δεύτερο κώδικα επειδή του έχουμε δομήσει την αρχιτεκτονική με διασυνδέσεις σημάτων και με στιγμιότυπα εξαρτημάτων με τη χρήση περιγραφής υλικού θα μας συνθέσει ακριβώς το ίδιο κύκλωμα με την εικόνα **4**.

## Κώδικες σε VHDL

Παρακάτω παρατίθενται οι κώδικες της εργασίας:

---

### Mux\_2\_x\_1

---

```
--Dhlwsh bibliothhkhhs kai tun paketun poy tha xrhsimopoihsoume
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY mux_2_x_1 IS
  GENERIC(N: NATURAL :=8);
  PORT (
    -- Shma eisodoy poy lambanei mia akoloythia timun.
    a:IN std_logic_vector(N-1 downto 0);
    -- Shma eisodoy poy lambanei mia akoloythia timun.
    sel:IN std_logic_vector(N-7 downto 0);
    -- Shma eisodoy poy lambanei mia akoloythia timun.
    b :IN std_logic_vector(N-1 downto 0);
    -- Shma ejodoy poy lambanei mia akoloythia timun.
    x :OUT std_logic_vector(N-1 downto 0));
  END mux_2_x_1;

--Dhlwsh arxitektonikhhs
ARCHITECTURE behavioral OF mux_2_x_1 IS

BEGIN
  --Ylopoihs toy pinaka alhtheias
```

```

--dld analoga me thn timh toy sel
--pairnei kai antistoixh timh toy pinaka to shma ejodoy
    x<= "00000000" WHEN sel = "00" ELSE
    a WHEN sel = "01" ELSE
    b WHEN sel = "10" ELSE
    "ZZZZZZZZ" WHEN sel = "11";

END behavioral;

```

---

## Second\_Mux\_2\_x\_1

---

```

--Dhlwsh bibliothhkhhs kai tum paketum poy tha xrhsimopoihsyme
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

ENTITY second_mux_2_x_1 IS
GENERIC(N: NATURAL :=8);
PORT (
-- Shma eisodoy poy lambanei mia akoloythia timun.
    a:IN std_logic_vector(N-1 downto 0);
-- Shma eisodoy poy lambanei mia akoloythia timun.
    sel:IN std_logic_vector(1 downto 0);
-- Shma eisodoy poy lambanei mia akoloythia timun.
    b :IN std_logic_vector(N-1 downto 0);
-- Shma ejodoy poy lambanei mia akoloythia timun.
    x :OUT std_logic_vector(N-1 downto 0));
END second_mux_2_x_1;

--Dhlwsh arxitektonikhhs
ARCHITECTURE behavioral OF second_mux_2_x_1 IS
BEGIN
--Ylopoihs toy pinaka alhtheias
--dld analoga me thn timh toy sel
--pairnei kai antistoixh timh toy pinaka to shma ejodoy
    WITH (sel) SELECT
        x <= "00000000" WHEN "00",
        a WHEN "01",
        b WHEN "10",
        "ZZZZZZZZ" WHEN "11",
        "-----" WHEN others;

END behavioral;

```

---

## NOT\_GATE

---

```
--Dhlwsh bibliothhkhhs kai tun paketun poy tha xrhsimopoihsoyme
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

--Dhlwsh ontothtas
ENTITY NOT_GATE IS
PORT (
  -- Shma eisodoy poy lambanei ena bit.
  A : IN BIT;
  -- Shma ejodoy poy lambanei ena bit.
  Y : OUT BIT
);
END NOT_GATE;

--Dhlwsh arxitektonikhhs
ARCHITECTURE behavioral OF NOT_GATE IS

BEGIN

  --Logikh prajh NOT
  Y <= not A;

END behavioral;
```

---

## AND\_GATE

---

```
--Dhlwsh bibliothhkhhs kai tun paketun poy tha xrhsimopoihsoyme
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

--Dhlwsh ontothtas
ENTITY AND_GATE IS
PORT (
  -- Shma eisodoy poy lambanei ena bit.
  A : IN BIT;
  -- Shma eisodoy poy lambanei ena bit.
  B : IN BIT;
  -- Shma ejodo poy lambanei ena bit.
  Y : OUT BIT
);
END AND_GATE;
```

```

--Dhlwsh arxitektonikhs
ARCHITECTURE behavioral OF AND_GATE IS

BEGIN
  --Logikkh prajh AND
  Y <= (A AND B);

END behavioral;

```

---

## OR\_GATE

---

```

--Dhlwsh bibliothhkhhs kai tun paketun poy tha xrhsimopoihsyme
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

--Dhlwsh ontothtas
ENTITY OR_GATE IS
PORT (
  -- Shma eisodoy poy lambanei ena bit.
  A : IN BIT;
  -- Shma eisodoy poy lambanei ena bit.
  B : IN BIT;
  -- Shma ejodo poy lambanei ena bit.
  Y : OUT BIT
);
END OR_GATE;

--Dhlwsh arxitektonikhs
ARCHITECTURE behavioral OF OR_GATE IS

BEGIN
  --logikkh prajh OR
  Y <= (A OR B);

END behavioral;

```

---

## Circuit-A

---

```
--Dhlwsh bibliothhkhhs kai tun paketun poy tha xrhsimopoihsoyme
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

--Dhlwsh ontothtas
ENTITY circuitA IS
PORT (
  -- Shma eisodoy poy lambanei ena bit.
  A : IN BIT;
  -- Shma eisodoy poy lambanei ena bit.
  B : IN BIT;
  -- Shma eisodo poy lambanei ena bit.
  C : IN BIT;
  -- Shma ejodo poy lambanei ena bit.
  Z : OUT BIT
);
END circuitA;

ARCHITECTURE behavioral OF circuitA IS

BEGIN
  --akoloythiakh ektelesh toy parakatw blok kwдика
  PROCESS(A,B,C)

    BEGIN
      --ean isxuei mia apo tis dyo h kai oi dyo
      IF ((A AND B) = '1' OR C = '1') THEN
        --tote these to shma ejodoy iso me '1'
        Z<='1';
      ELSE
        --Se opoiadhpote allh periptwsh these to shma
        --ejodoy iso me '0'
        Z<='0';
      END IF;
    END PROCESS;
  END behavioral;
```

---

## Circuit-B

---

```
--Dhlwsh bibliothhkhhs kai tun paketun poy tha xrhsimopoihsoume
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;

--Dhlwsh ontothtas
ENTITY circuitB IS
PORT (
-- Shma eisodoy poy lambanei ena bit.
    A : IN BIT;
-- Shma eisodoy poy lambanei ena bit.
    B : IN BIT;
-- Shma eisodoy poy lambanei ena bit.
    C : IN BIT;
-- Shma ejodo poy lambanei ena bit.
    Z : OUT BIT
);
END circuitB;

--Dhlwsh arxitektonikhhs
ARCHITECTURE structural OF circuitB IS

--Klhsh toy stoixeio NOT_GATE kai dhlwsh eisodwn ejodwn
    COMPONENT NOT_GATE
        PORT(A : IN BIT; Y : OUT BIT);
    END COMPONENT;

--Klhsh toy stoixeio AND_GATE kai dhlwsh eisodwn ejodwn
    COMPONENT AND_GATE
        PORT(A,B : IN BIT; Y : OUT BIT);
    END COMPONENT;

--Klhsh toy stoixeio OR_GATE kai dhlwsh eisodwn ejodwn
    COMPONENT OR_GATE
        PORT(A,B : IN BIT; Y : OUT BIT);
    END COMPONENT;

--Dhlwsh prosurinwn metablhtwn(endiamesa shmata)
SIGNAL x1,x2,x3,x4 : BIT;

--Dhmiourgia toy kyklwmatos me bash thn eikona 4 ths anaforas
--Xtizoume to kyklwma dhladh me endiamesa shmata kai eidh
--stoixeia ylikoy ta opoia exoume sthn diathesh mas.
```

```

BEGIN
    GATE_1 : NOT_GATE PORT MAP (C,x2);--C eisodos,x2 ejodos
    GATE_2 : AND_GATE PORT MAP (A,B,x1);--A,B eisodoi,x1 ejodos
    GATE_3 : AND_GATE PORT MAP (x1,x2,x3);--x1,x2 eisodoi,x3 ejodos
    GATE_4 : OR_GATE PORT MAP (x1,x3,x4);--x1,x3 eisodos,x4 ejodos
    GATE_5 : OR_GATE PORT MAP (C,x4,Z);--C,x4 eisodos,Z ejodos
END structural;

```